A Hybrid Algorithm for Coverage Path Planning With Imperfect Sensors

Michael Morin[1,✉] Irène Abi-Zeid[2] Yvan Petillot[3] Claude-Guy Quimper[1]

***Abstract*— We are interested in the *coverage path planning* problem with imperfect sensors, within the context of robotics for mine countermeasures. In the studied problem, an *autonomous underwater vehicle* (AUV) equipped with sonar surveys the bottom of the ocean searching for mines. We use a cellular decomposition to represent the ocean floor by a grid of uniform square cells. The robot scans a fixed number of cells sideways with a varying probability of detection as a function of distance and of seabed type. The goal is to plan a path that achieves the minimal required coverage in each cell while minimizing the total traveled distance and the total number of turns. We propose an off-line hybrid algorithm based on dynamic programming and on a traveling salesman problem reduction. We present experimental results and show that our algorithm's performance is superior to published results in terms of path quality and computational time, which makes it possible to implement the algorithm in an AUV.**

I. Introduction

We are interested in the *coverage path planning* (CPP) problem in the context of deploying mobile autonomous underwater vehicles (AUVs) for mine countermeasures (MCM). Our specific application consists of planning the path of an AUV from the REMUS family of vehicles [1], designed to swim long distances at constant speed and altitude with infrequent turns, and with no or very little capacity to sense and avoid obstacles [2]. Our AUV is fitted with a navigation system to adjust its position. It is also equipped with sidescan sonar to search for mines on the bottom of the ocean [3]. Without loss of generality, we assume that there are no blind spots, since the use of "gap filler" forward looking sonars has become prevalent. We assume imperfect sensors where the conditional probability of detecting a target of interest given that it is within the sensor's range is less than 100% [4], imperfect sensitivity and perfect specifity (no false positives). This conditional probability of detection can be interpreted as the degree of coverage resulting from sensor or actuator imperfectness [5]. While traveling on a path segment, the AUV surveys a fixed distance sideways with a varying conditional probability of detection that depends on the seabed type of the surveyed region (e.g., complex seabed, sand ripples, flat seabed), and on the range of the sensor. A given minimal coverage (minimal conditional probability of detection) must be achieved over the whole area of interest for the path to be *feasible*.

CPP problems are often solved in order to plan an agent's (or multiple agents) path in such a way to guarantee complete coverage [6], or in the case of imperfect sensors, a minimal required coverage [7]. In complete CPP problems with perfect sensors, a cell is fully covered after a single scan and no further visits are needed. Complete CPP problems (also known as area covering, or region filling) arise in many practical applications, e.g., minesweeping [8], [9], seabed surveys in harbors and waterways [2], robotic mowing, harvesting and ploughing in agricultural applications [10], marine habitat planning [11], and floor cleaning [12].

A CPP problem is different from classical path planning problems where the robot moves from an initial point to a known destination. It is also different from optimal detection search problems for search and rescue or surveillance [13], [14]. In detection search theory, the goal is often to plan a path that maximizes the global probability of finding a search object (e.g., survivor, crashed plane, lost vessel). In these problems, the search stops after detection and the probability distribution of the whereabouts of the search object is often known a priori. In most CPP problems, the objective is to attain a complete coverage of the whole environment. Furthermore, in CPP, the whereabouts of objects locations are often unknown, although some authors assume a priori knowledge of targets locations [15], [16], [17].

Time to completion is an important issue. Longer paths to cover an area take more time. Turning also takes more time and may increase navigational errors [18]. Therefore, the goal is to find a feasible path that minimizes the total traveled distance and the total number of turns.

This paper is organized as follows. We discuss related work in section II. Section III formalizes our problem in the context of underwater minesweeping operations. Section IV describes our novel hybrid algorithm based on *dynamic programming* and on a *traveling salesman problem* (TSP) reduction. Section V presents, compares and discusses our experimental results. We conclude in section VI.

II. Related Work

Most path planning algorithms use a discretized representation of the continuous environment. In a *cellular decomposition*, the continuous environment is divided in a set of uniform or non-uniform cells [19]. *Uniform cellular decompositions* involve grids where cells have the same size. If the environment contains obstacles, we have an occupancy

*This work was partially funded by the Natural Sciences and Engineering Research Council of Canada and by the Fonds de recherche du Québec-Nature et technologies.

[1]M. Morin and C.G. Quimper are with the Department of Computer Science and Software Engineering, Université Laval, Québec, Qc, Canada ✉Michael.Morin.3 at ulaval.ca

[2]I. Abi-Zeid is with the Department of Operations and Decision Systems, Université Laval, Québec, Qc, Canada

[3]Y. Petillot is with the Ocean Systems Laboratory, Heriot-Watt University, Edinburgh, Scotland, UK

grid [20]. Such a decomposition technique is considered to be *approximate* since some cells may be partially obstructed and some parts of the environment may not be fully covered.

In a *non-uniform cellular decomposition* the constraint on the cell size is relaxed. These include trapezoidal decomposition, boustrophedon decomposition (both exact), and recursive cellular decomposition (approximate). In a CPP, when the cellular decomposition results in cells that are larger than the range of the sensor, a lawnmower pattern is often assumed in the cells [18]. Depending on the discretization scale and on the range of the sensor, whether limited to its circumference, extended, or infinite, distant cells may or may not be surveyed. When the environment is known *a priori*, the problem may be formulated as an *off-line CPP* problem. Otherwise, we have an *on-line CPP* problem where the agent must discover its environment. For on-line CPP problems a *sensor-based approach* [18] is used, i.e., the robot uses its sensors to acquire knowledge on its environment. In our case, since the seabed map is available as a grid of uniform square cells, we have an off-line CPP problem where the discretization scale is such that the side of a cell is not larger than the sonar's range.

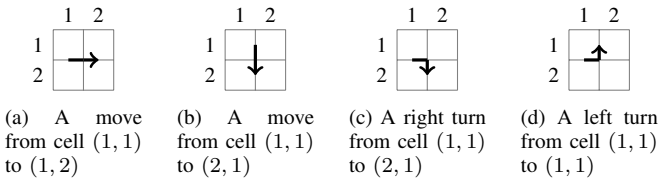
There exists a large body of literature on path planning in robotics, mostly for ground robotics in four general areas: navigation, coverage, localization and mapping. CPP methods and path planning for navigation are reviewed in [18] and [21] respectively.

The problem presented in this paper differs from classical CPP problems in three ways: First, the sensors are imperfect; and second, the range of detection is not limited to the location of the AUV and varies with the distance separating it from the scanned area, and with the seabed type. Third, we do not require perfect coverage, we rather must ensure a minimum coverage level everywhere in the area of interest. We call this formulation, the *CPP with imperfect extended detection* ($CPPIED$). To our knowledge, the only published work on this formulation is that of [7]. He presented the advantages and disadvantages of existing approaches and concluded that none of them was suitable for the $CPPIED$. He proposed the *heterogeneous coverage path planning* (HCPP) algorithm for planning an AUV's path, a highly instance dependent heuristic that must be fine-tuned in a trial and error fashion.

As the $CPPIED$ problem deals with a required coverage in all cells, we also need to distinguish it from the *multirobot-controlled frequency coverage* ($MRCFC$) problem [22]. The two problems are similar in that many visits will be needed in each location (cell). However, the goal of the former is to achieve the required coverage in each cell whereas the goal of the latter is to be as close as possible to the prescribed relative frequency of visits.

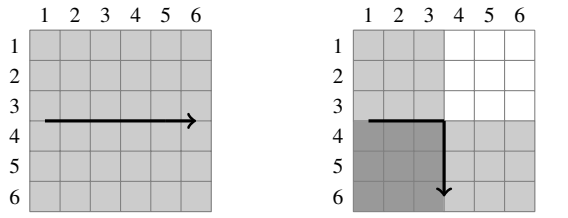
III. The *CPPIED* Problem

Let $\mathcal{T}$ be the set of seabed types, for example, flat or ripple sand. The ocean is represented by a $m \times n$ matrix $\mathbf{O}$ such that $\mathbf{O}_{ij} \in \mathcal{T}$ is the seabed type in the cell (i, j). The cell $(1, 1)$ is located in the upper left corner of the grid. The position of the robot is defined by $((i, j), dir)$ where (i, j) is a grid cell and $dir \in \{\text{north}, \text{south}, \text{east}, \text{west}\}$ is the direction in which the robot is pointing. The robot moves on the grid lines between the cells. A robot pointing north or south in cell (i, j) is located in the middle of the vertical line between cells (i, j) and $(i, j + 1)$. A robot pointing east or west in cell (i, j) is located in the middle of the horizontal line between cells (i, j) and $(i + 1, j)$. The robot moves forward one cell at a time (Fig. 1a and 1b). It cannot stop, backup, nor turn around on the spot. However, $90°$ turns while moving forward are allowed (Fig. 1c and 1d). The robot scans $2r$ cells: r on its left, r on its right. No diagonal scanning occurs while the robot is turning. However, some cells may be scanned a second time just after the turn. Two possible paths and their set of scanned cells are shown on Fig. 2 for range $r = 3$.



(a) A move from cell $(1, 1)$ to $(1, 2)$ (b) A move from cell $(1, 1)$ to $(2, 1)$ (c) A right turn from cell $(1, 1)$ to $(2, 1)$ (d) A left turn from cell $(1, 1)$ to $(1, 1)$

Fig. 1. Feasible moves on a uniform grid



(a) A path from cell $(3, 1)$ to cell $(3, 6)$ with $r = 3$ (b) A path from cell $(3, 1)$ to cell $(6, 3)$ with $r = 3$

Fig. 2. Scans on a uniform seabed; light gray shaded cells are scanned once, and dark gray shaded cells are scanned twice.

The conditional detection probability of a sensor scan in a cell (i, j) (given that a mine is present) is a function $p^{\text{scan}} : \mathbb{N}^+ \times \mathcal{T} \to [0.0, 1.0]$ of the distance $d(x, y, i, j)$, in number of cells, between the current robot's cell (x, y) and the scanned cell (i, j), and of the seabed type $\mathbf{O}_{ij}$ of the scanned cell (i, j). For instance, with $\mathcal{T} = \{$flat (f), ripples (r), complex (c)$\}$ and $r = 3$, the sensor's conditional detection probabilities are expressed as:

$$p^{\text{scan}} = \begin{bmatrix} p^{\text{scan}}(1,\text{c}) & p^{\text{scan}}(2,\text{c}) & p^{\text{scan}}(3,\text{c}) \\ p^{\text{scan}}(1,\text{r}) & p^{\text{scan}}(2,\text{r}) & p^{\text{scan}}(3,\text{r}) \\ p^{\text{scan}}(1,\text{f}) & p^{\text{scan}}(2,\text{f}) & p^{\text{scan}}(3,\text{f}) \end{bmatrix}. \quad (1)$$

The current coverage map of a grid environment is represented by a $m \times n$ matrix $\mathbf{C}$ where $\mathbf{C}_{ij}$ is the achieved conditional detection probability in the cell (i, j). The initial coverage map is the null matrix. After a first scan of cell (i, j) from cell (x, y), the coverage in (i, j) is $\mathbf{C}_{ij} = p^{\text{scan}}(d(x, y, i, j), \mathbf{O}_{ij})$. The conditional probability of non-detection (i.e., the miss probability, or the non-coverage) is $1 - \mathbf{C}_{ij} = 1 - p^{\text{scan}}(d(x, y, i, j), \mathbf{O}_{ij})$. Given independent detections and following a scan in cell (i, j)

from cell (x', y'), the non-coverage in cell (i, j) is $1 - \mathbf{C}'_{ij} = (1 - \mathbf{C}_{ij})(1 - p^{\text{scan}}(d(x', y', i, j)), \mathbf{O}_{ij})$. Therefore, the updated coverage of cell (i, j) is $\mathbf{C}'_{ij} = 1-(1-\mathbf{C}_{ij})(1-p^{\text{scan}}(d(x', y', i, j)), \mathbf{O}_{ij})$ leading to the update equation (2).

$$\mathbf{C}'_{ij} \coloneqq \mathbf{C}_{ij} + (1 - \mathbf{C}_{ij})p^{\text{scan}}(d(x', y', i, j), \mathbf{O}_{ij}). \quad (2)$$

The required coverage is represented by a $m \times n$ matrix $\mathbf{D}$ such that $\mathbf{D}_{ij}$ represents the required minimum coverage that must be attained in cell (i, j). The required coverage is achieved when $\mathbf{D} \leq \mathbf{C}$.

A *CPPIED* problem instance is defined as a tuple $\left(\mathcal{T}, \mathbf{O}, \mathbf{D}, p^{\text{scan}}, (i^{\text{init}}, j^{\text{init}})\right)$ where $(i^{\text{init}}, j^{\text{init}})$ is the initial location cell of the robot.

IV. A Hybrid Algorithm Based on Dynamic Programming and the Traveling Salesman

The goal of the proposed algorithm is to define, in lexicographic order, a feasible shortest path (the first objective), consisting of segments, that also minimizes the number of turns (the second objective). A segment is a set of horizontally or vertically adjacent cells that does not contain any turns. The algorithm is composed of two main phases. In the first phase, we construct a set $\mathcal{S}$ of disconnected segments such that a robot traveling along all these segments will achieve the required coverage $\mathbf{D}$. In the second phase, we use a *TSP* reduction to optimally connect the segments obtained in phase 1 and form the desired path P. This is similar to the *TSP* formulation used in [2] to connect paths in sub-areas of a harbor environment. Our algorithm proceeds as follow:

1) Initialize the robot's path P to an empty path, the set of segments $\mathcal{S}$ used to construct path P to the empty set, and the current coverage matrix $\mathbf{C}$ to the null matrix. At each iteration, construct and choose a set of horizontal or a set of vertical segments that maximizes coverage gains (rewards) ($\mathcal{H}^*$ or $\mathcal{V}^*$) and add it to $\mathcal{S}$ (Section IV-A). Iterate until the required coverage is attained by the segments of $\mathcal{S}$, i.e., $\mathbf{D} \leq \mathbf{C}$.
2) Link the segments of $\mathcal{S}$ in an optimal fashion to obtain a feasible path P of minimal length (and minimal number of turns) (Section IV-B).

The aim in phase 1 is to construct a segment set $\mathcal{S}$ allowing the robot to achieve the required coverage with the shortest possible path. Therefore, we need short segments. Intuitively, a set $\mathcal{S}$ with a small cardinality is desirable as well since it will lead to a lower number of turns in phase 2.

A. Greedily Choosing a Set of Segments $\mathcal{S}$

Some difficulties in constructing a segment set $\mathcal{S}$ arise due to the extended sensor's range since multiple detections in a given cell may arise from two different segments or more. To overcome this difficulty, we impose, at each iteration, a $2r$ spacing constraint within the set $\mathcal{H}^*$ of horizontal segments and the set $\mathcal{V}^*$ of vertical segments. Therefore, the rewards of the parallel segments are independent, which allows us to use polynomial time dynamic programming algorithms to compute both the reward of a robot traveling on a segment, and the optimal sets of segments $\mathcal{H}^*$ and $\mathcal{V}^*$.

In order to compute the horizontal gain in cell (i, j), $\mathbf{H}_{ij}$, let $p^{\text{C}}(i, j, k)$ be the updated probability of detection when a cell (i, j) is scanned from a distance of k cells:

$$p^{\text{C}}(i, j, k) = \mathbf{C}_{ij} + (1 - \mathbf{C}_{ij})p^{\text{scan}}(k, \mathbf{O}_{ij}). \quad (3)$$

The gain obtained by scanning cell (i, j) from distance k can be defined as:

$$g(i, j, k) = \begin{cases} \min\{\mathbf{D}_{ij}, \\ \qquad p^{\text{C}}(i, j, k)\} - \mathbf{C}_{ij} & \mathbf{C}_{ij} < \mathbf{D}_{ij}; \\ -\lambda & \mathbf{C}_{ij} \geq \mathbf{D}_{ij}. \end{cases} \quad (4)$$

If cell (i, j) is not already covered, i.e., $\mathbf{C}_{ij} < \mathbf{D}_{ij}$, the gain equals the increment in probability of detection in (i, j) up to the required probability of detection value, i.e., $\min\left\{\mathbf{D}_{ij}, p^{\text{C}}(i, j, k)\right\} - \mathbf{C}_{ij}$. Otherwise, cell (i, j) is covered, i.e., $\mathbf{C}_{ij} \geq \mathbf{D}_{ij}$, and we impose a small penalty λ for overcoverage. Let $G(i, j)$ be the sum of the gains, in probability of detection, in the cells within the range of the robot:

$$G(i, j) = \sum_{k=1}^{r} g(i - k + 1, j, k) + \sum_{k=1}^{r} g(i + k, j, k). \quad (5)$$

Since we wish to find the shortest possible segment with the highest coverage, the segment ends before the first cell which gain $G(i, j)$ is less than or equal to zero. Therefore, the horizontal gain of a cell (i, j), $\mathbf{H}_{ij}$, is defined as follows:

$$\mathbf{H}_{ij} = \begin{cases} G(i, j) & G(i, j) > 0; \\ -\infty & G(i, j) \leq 0. \end{cases} \quad (6)$$

The endpoints a and b that define the segment h_i such that the sum of its gains is maximal are identified for each row i as follows:

$$K(h_i) = \max_{a,b} \sum_{j=a}^{b} \mathbf{H}_{ij}. \quad (7)$$

This problem, called the *Maximum subarray* problem, is solvable in linear time using Kadane's algorithm [23], a simple example of dynamic programming.

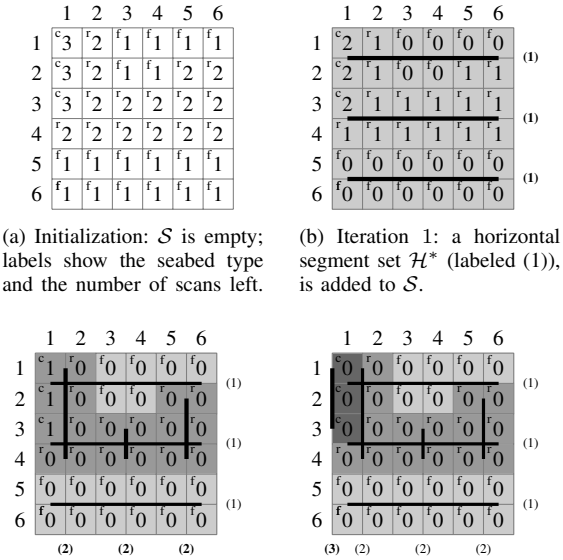
The optimal horizontal set $\mathcal{H}^*$ is the subset of segments that maximizes the sum of the horizontal gains over the rows subject to a $2r$ spacing constraint:

$$\max_{I \subset \{1..m\} | i, i' \in I \Rightarrow |i - i'| \geq 2r} \sum_{i \in I} K(h_i). \quad (8)$$

We compute $\mathcal{H}^*$ using the following recurrence relation:

$$\mathbf{h^*}_i = \begin{cases} K(h_i) & 1 \leq i \leq 2r; \\ \max\{\mathbf{h^*}_{i-1}, & \\ \qquad K(h_i) + \mathbf{h^*}_{i-2r}\} & 2r < i \leq m, \end{cases} \quad (9)$$

where $\mathbf{h^*}_i$ is the maximum gain achieved by a robot when it travels along segment h_i. With this technique, choosing an optimal set of segments under a $2r$ spacing constraint is done in polynomial time. The set $\mathcal{V}^*$ is computed in the same fashion following a map rotation of $90°$. The set among $\mathcal{H}^*$ or $\mathcal{V}^*$ providing the highest coverage is added to $\mathcal{S}$. Ties are broken using the set with the smallest cardinality. Note that there is no spacing constraint in $\mathcal{S}$.



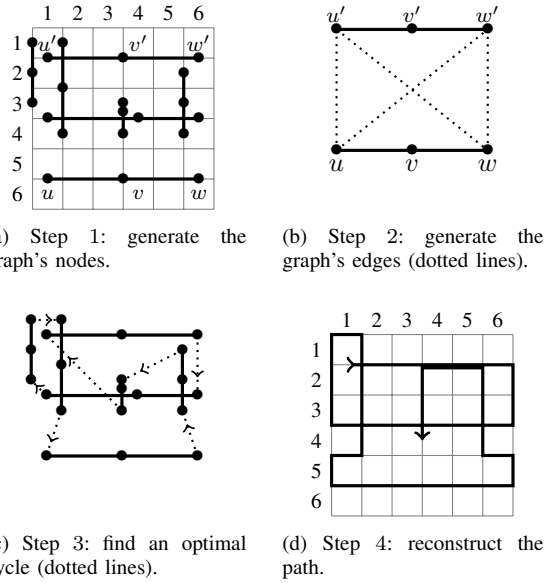
(a) Initialization: $\mathcal{S}$ is empty; labels show the seabed type and the number of scans left.

(b) Iteration 1: a horizontal segment set $\mathcal{H}^*$ (labeled (1)), is added to $\mathcal{S}$.

(c) Iteration 2: a vertical segment set $\mathcal{V}^*$ (labeled (2)) is added to $\mathcal{S}$.

(d) Iteration 3: a vertical segment set $\mathcal{V}^*$ (labeled (3)) is added to $\mathcal{S}$.

Fig. 3. The construction of the segment set $\mathcal{S}$ on a map with $\mathcal{T} = \{$flat (f), ripples (r), complex (c)$\}$, and $r = 1$; dark gray shaded cells were scanned three times, medium gray shaded cells were scanned twice, light gray shaded cells were scanned once.



(a) Step 1: generate the graph's nodes.

(b) Step 2: generate the graph's edges (dotted lines).

(c) Step 3: find an optimal cycle (dotted lines).

(d) Step 4: reconstruct the path.

Fig. 4. The TSP reduction and the path reconstruction on the segments set $\mathcal{S}$ of Fig. 3.

1) Example: Fig. 3 presents an example for choosing an optimal segment set on a 6×6 grid with $r = 1$. In each cell of the grid, we indicate the number of robot scans needed to achieve the required coverage $\mathbf{D}$. This number is an upper bound that depends on the seabed type, on the current coverage matrix $\mathbf{C}$, and on the probability of detection p^{scan}. The set $\mathcal{S}$ is initialized to the empty set (Fig. 3a), and $\mathbf{C} = 0$. At iteration 1 (Fig. 3b), the procedure generates a first set $\mathcal{H}^*$ of horizontal segments and a first set $\mathcal{V}^*$ of vertical segments. $\mathcal{H}^*$ is then added to $\mathcal{S}$ ($\mathcal{V}^*$ is equivalent in terms of gain, in this case). At iteration 2 (Fig. 3c), the updated current coverage matrix $\mathbf{C}$ is used to compute two new sets $\mathcal{H}^*$ and $\mathcal{V}^*$. The algorithm adds $\mathcal{V}^*$ to $\mathcal{S}$ since it achieves a higher gain. At iteration 3 (Fig. 3d), the algorithm chooses $\mathcal{V}^*$ (a single segment) and adds it to $\mathcal{S}$.

B. Reducing to a TSP and Reconstructing the Path P

In this phase, a connected path P including all the segments in $\mathcal{S}$ is constructed using a TSP reduction. Let $\mathcal{V}(G)$ be the set of nodes of graph G, $\mathcal{E}(G)$ be its set of edges, and $c(e)$ be a cost function on $\mathcal{E}(G)$. The goal is to find a cycle of minimal cost that visits each node once.

For each segment s_i in $\mathcal{S}$ with endpoints u_i and w_i, we create three nodes in $\mathcal{V}(G)$: u_i, w_i, and v_i and two edges (u_i, v_i) and (v_i, w_i). The node v_i is a dummy node used to force the algorithm to travel over the segment between u_i and w_i. The cost of these two edges is null. For every pair of segments s_i and s_j in $\mathcal{S}$, we create four edges: (u_i, u_j), (u_i, w_j), (w_i, u_j), and (w_i, w_j). The cost of these edges corresponds to the shortest distance in the number of robot's moves needed to connect the endpoints. Finally, let u^s be the starting position of the robot. We create a source node u^s connected to all endpoints of the segments with distances given by the shortest distance in number of moves. We create a sink node w^s connected to all segments' endpoints with a null distance. We create a dummy node v^s that is connected to u^s and w^s by two edges of null cost.

Solving the TSP consists of finding a cycle of minimum cost that passes through each node exactly once. Note that such a cycle would have to visit each node v_i once. Since this node is only connected to the two endpoints u_i, w_i, it forces the cycle to enter by one endpoint of the segment and to leave by the other endpoint. The same principle applies for the node v^s that is only connected to the source node u^s and the sink node w^s. Therefore, a cycle starts at node w^s, visits all the segments, and returns to w^s through v^s. Removing the node w^s and v^s from the cycle yields the solution path P we are looking for. To find the lowest cost cycle, and therefore the solution path, we chose the Concorde solver [24].

1) Example: Fig. 4 shows a TSP reduction and a path reconstruction example on the segments set $\mathcal{S}$ of Fig. 3. First, the reduction process generates the graph nodes represented by dots on Fig. 4a. The robot starts in cell $(1, 1)$. Therefore, the source node u^s is positioned in $(1, 1)$, the sink node w^s and the dummy node v^s are positioned at the same place. Second, it links the segments endpoints by creating edges. Fig. 4b shows the edges added to the graph G for segments $s = (u, v, w)$ and $s' = (u', v', w')$. Third, on Fig. 4c, the Concorde solver finds an optimal cycle that starts from the source and goes through all the nodes before returning to the source node u^s by the sink node w^s then the dummy node v^s. Finally, on Fig. 4d, the robot's path is reconstructed. As shown, there are no links between the last robot's position $(4, 3)$ and its initial position in $(1, 1)$: We simply ignore the sink and the dummy nodes during path reconstruction.

TABLE I

PROBLEM INSTANCES PUBLISHED IN [7]

No	Ocean bed $\mathbf{O}$	Required coverage $\mathbf{D}_{ij}$ $(\forall i,j)$	Detection prob. p^{scan}
1	[7, Fig. 5.1]	0.9	$\begin{bmatrix} 0.6 & 0.65 & 0.62 \\ 0.7 & 0.85 & 0.8 \\ 0.99 & 0.99 & 0.99 \end{bmatrix}$
2	[7, Fig. 5.4]	0.9	$\begin{bmatrix} 0.7 & 0.8 & 0.75 \\ 0.99 & 0.99 & 0.99 \\ 0.91 & 0.95 & 0.92 \end{bmatrix}$
3	[7, Fig. 5.8]	0.9	$\begin{bmatrix} 0.8 & 0.8 & 0.8 \\ 0.8 & 0.8 & 0.8 \\ 0.91 & 0.91 & 0.91 \end{bmatrix}$
4	[7, Fig. 5.11]	0.75	$\begin{bmatrix} 0.51 & 0.51 & 0.51 \\ 0.8 & 0.8 & 0.8 \\ 0.91 & 0.91 & 0.91 \end{bmatrix}$
5	[7, Fig. 5.14]	0.75	$\begin{bmatrix} 0.51 & 0.51 & 0.51 \\ 0.8 & 0.8 & 0.8 \\ 0.99 & 0.99 & 0.99 \end{bmatrix}$
6	[7, Fig. 5.16]	0.9	$\begin{bmatrix} 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 \\ 0.91 & 0.91 & 0.91 \end{bmatrix}$
7	[7, Fig. 5.18]	0.85	$\begin{bmatrix} 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 \\ 0.91 & 0.91 & 0.91 \end{bmatrix}$

V. EXPERIMENTATION

We present the results of our DpSweeper algorithm, implemented in C++, obtained on an Intel(R) Core(TM) i7-Q740 CPU with 8 GB of RAM. A comparison with the results of the HCPP algorithm on seven instances published in [7] is included. Table I summarizes the instances in increasing order of complexity. The robot starts in the top left corner of the grid. The required coverage matrix $\mathbf{D}$ is uniform, i.e., all cells have the same required coverage probability. We use a range $r = 3$. The seabed maps $\mathbf{O}$, containing more than 21000 cells, can consist of three different seabed types: flat (f), ripples (r), and complex (c) seabed. The first instance [7, Fig. 5.1] has a flat seabed: a lawnmower pattern is sufficient to cover it. The second [7, Fig. 5.4] is made of a flat seabed with a rectangular complex seabed patch in the middle. The third [7, Fig. 5.8] has three complex seabed patches. The fourth [7, Fig. 5.11] has a circle of complex seabed in the middle. The fifth [7, Fig. 5.14] has a fragmented circle of complex seabed in the middle. The sixth [7, Fig. 5.16] is a real map containing three different seabed types. The seventh [7, Fig. 5.18] is a realistic randomly generated map containing three different seabed types.

Table II presents the results of both the HCPP and the DpSweeper algorithms. As in [7], we use the following lexicographic order of criteria: First minimize path length, then the number of turns. The required coverage is attained in all cases. We see that the DpSweeper algorithm outperforms the HCPP algorithm on the first criterion in all instances except for instances 2 and 5. Following a close inspection of the figures published in [7], we noticed that on the spot $180°$ turns were allowed. Our more restrictive assumptions (not allowing on the spot $180°$ turns), although not a DpSweeper limitation, are closer to the physical constraints of AUVs and actually favor HCPP. This may explain the slightly longer path we found on instances 2 and 5. For the instances 6 and 7, our solution has significantly more turns than HCPP. On these instances, generating more turns was necessary to diminish the path length of respectively more than 15% and 20% when compared to [7], which is coherent with the lexicographic order of the minimization objectives.

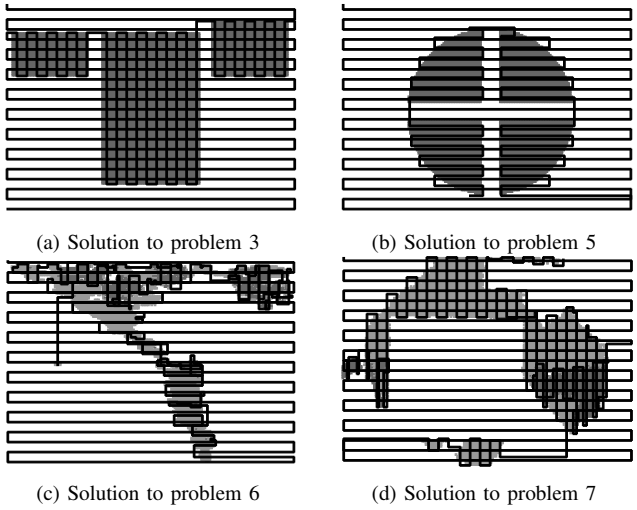
TABLE II

COMPARISON OF DPSWEEPER TO HCPP [7]

	HCPP†		***DpSweeper***				
	Solution‡		Solution‡		TSP	Time (s)	
No	Length	Turns	Length	Turns	nodes	TSP	Total
1	3777	40	**3755**	41	66	< 1	< 1
2	**4599**	**60**	4628	61	99	< 1	1
3	5556	82	**5321**	95	144	< 1	1
4	5494	77	**5363**	**75**	117	< 1	1
5	**5119**	114	5151	107	162	1	2
6	6689	136	**5681**	270	402	45	52
7	7141	137	**5731**	174	273	12	15

† The solving times of the HCPP method are not published.

‡ Shorter is better, ties are broken on the number of turns.



(a) Solution to problem 3 (b) Solution to problem 5

(c) Solution to problem 6 (d) Solution to problem 7

Fig. 5. Solutions found by DpSweeper; complex, ripples, and flat seabed cells are respectively filled with dark gray, gray, and white.

The DpSweeper algorithm solved instances 1 to 4 in one second or less and instance 5 in two seconds. The solving times of the most realistic instances (6 and 7) are within a minute. These times include both the segment generation process and the Concorde solver calls. We chose to use Concorde instead of a TSP heuristic since it is the state of the art: It solves most TSPLIB [25] instances (sometimes with more than 2000 nodes) within a few minutes. Although Concorde may consume more time than a heuristic, it finds the optimal tour to all our instances within a few seconds.

Fig. 5 shows some of the solutions provided by our algorithm. On the problem 3 (Fig. 5a), we see the tendency of the algorithm to generate a simple lawnmower pattern that minimizes both the path length and the number of turns. It starts with a horizontal lawnmower pattern. Then, on the third row, it follows a vertical lawnmower to cover the complex seabed patches. Finally, it comes back to its

horizontal lawnmower pattern to cover all the grid. We notice a similar behavior on problem 5 (Fig. 5b). The seabed of this grid forces the algorithm to adapt the length of each segment to follow the fragmented circle edges. For problem 6 (Fig. 5c), the algorithm first plans a complete lawnmower pattern. Then, it goes up towards and over the closest ripples seabed and the complex patches. Finally, it passes over complex seabed patches again to achieve the required coverage, and then aims for the furthest ripples patches. It ends its course on the left hand side of the map. A similar behavior occurs in Fig. 5d.

In addition to its superior results, the main advantage DpSweeper over HCPP is that it is general, and contrary to HCPP, does not require lengthy, instance specific, fine-tuning. Furthermore, its very short computational time makes it possible to obtain a high quality path rapidly, an important characteristic for algorithms in practical contexts.

VI. Conclusion

We have presented a novel algorithm (DpSweeper) for coverage path planning using imperfect sensors with an extended detection range ($CPPIED$). In order to obtain shortest feasible paths with small numbers of turns, the algorithm consists of two main phases: (1) greedily constructing a partial path made of segments to guarantee that the required coverage is achieved (dynamic programming); (2) optimally linking segments to create a path that is within the robot's physical constraints (TSP reduction). The algorithm yields favorable results in a very short time compared to the literature. It is flexible and can be applied to general complex seabed environments. In contrast with the only other algorithm that tackles the $CPPIED$ problem, it does not require customized fine-tuning for individual environments. Even though the TSP problem is $\mathcal{NP}$-hard [26], the instances resulting from our real and practical $CPPIED$ problem instances are within the reach of the Concorde TSP solver that we used. TSP reductions are found in the literature to solve part of decomposed coverage problems (e.g., [2]) or other similar problems (e.g., [27]). However, most of them use the TSP to find a sequence of large regions where they assume a fixed coverage pattern. We use the TSP directly to order a sequence of path components (segments) on the whole environment. Our algorithm is not limited to an underwater context and may be used in other obstacle-free environments.

Acknowledgment

We wish to thank P. Giguère for his insightful suggestions, and the Ocean Systems Lab for providing us with the problem instances found in [7].

References

[1] J. Nicholson and A. Healey, "The present state of autonomous underwater vehicle (AUV) applications and technologies," *Marine Technology Society Journal*, vol. 42, no. 1, pp. 44–51, 2008.

[2] C. Fang and S. Anstee, "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle," in *Proc. of OCEANS 2010 IEEE-Sydney*, Sydney, Australia, May 2010, pp. 1–8.

[3] S. Reed, Y. Petillot, and J. Bell, "An automatic approach to the detection and extraction of mine features in sidescan sonar," *IEEE J. Oceanic Eng.*, vol. 28, no. 1, pp. 90–105, 2003.

[4] D. W. Gage, "Randomized search strategies with imperfect sensors," in *Proc. of SPIE Mobile Robots VIII*, San Diego, CA, Sept. 1993, pp. 270–279.

[5] ——, "Many-robot MCM search systems," in *Proc. of Autonomous Vehicles in Mine Countermeasures Symposium*, A. Bottoms, J. Eagle, and H. Bayless, Eds., Monterey, CA, Apr. 1995.

[6] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, AK, May 2010, pp. 5525–5530.

[7] M. Drabovich, "Automated mission trajectory planning for mine countermeasures operations," Master's thesis, Heriot Watt University, Edinburgh, Scotland, UK, 2008.

[8] D. P. Williams, "On optimal AUV track-spacing for underwater mine detection," in *Proc. of IEEE International Conference on Robotics and Automation (ICRA'10)*, Anchorage, AK, May 2010, pp. 4755–4762.

[9] J. R. Stack and C. M. Smith, "Combining random and data-driven coverage planning for underwater mine detection," in *Proc. of OCEANS 2003 MTS/IEEE*, vol. 5, San Diego, CA, Sept. 2003, pp. 2463–2468.

[10] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, 2009.

[11] E. Galceran and M. Carreras, "Coverage path planning for marine habitat mapping," in *Proc. of OCEANS 2012 MTS/IEEE*, Hampton Roads, VA, USA, Oct. 2012, pp. 1–8.

[12] R. N. De Carvalho, H. Vidal, P. Vieira, and M. Ribeiro, "Complete coverage path planning and guidance for cleaning robots," in *Proc. of IEEE International Symposium on Industrial Electronics (ISIE'97)*, vol. 2, Guimarães, Portugal, July 1997, pp. 677–682.

[13] M. Morin, L. Lamontagne, I. Abi-Zeid, P. Lang, and P. Maupin, "The optimal searcher path problem with a visibility criterion in discrete time and space," in *Proc. of the 12th International Conference on Information Fusion (FUSION'09)*. Seattle, WA: ISIF IEEE, July 2009, pp. 2217–2224.

[14] I. Abi-Zeid and J. R. Frost, "Sarplan: A decision support system for canadian search and rescue operations," *European Journal of Operational Research*, vol. 162, no. 3, pp. 630–653, 2005.

[15] B. Nguyen and D. Hopkin, "Modeling autonomous underwater vehicle (AUV) operations in mine hunting," in *OCEANS 2005 IEEE-Europe*, vol. 1, Brest, France, June 2005, pp. 533–538.

[16] E. Acar, Y. Zhang, H. Choset, M. Schervish, A. G. Costa, R. Melamud, D. Lean, and A. Graveline, "Path planning for robotic demining and development of a test platform," in *Proc. of the 8th International Conference on Field and Service Robotics (FSR 2012)*, Matsushima, Japan, July 2001, pp. 161–168.

[17] Y. Zhang, M. Schervish, E. U. Acar, and H. Choset, "Probabilistic methods for robotic landmine search," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, vol. 3, Maui, Hawaii, Oct./Nov. 2001, pp. 1525–1532.

[18] H. Choset, "Coverage for robotics," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[19] S. Russel and P. Norvig, *Intelligence artificielle*, 2nd ed. Paris, France: Pearson Education France, 2006.

[20] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *IEEE Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[21] L. Paull, S. Saeedi, and H. Li, "Path planning for autonomous underwater vehicles," in *Robot Autonomy*. Springer, 2013, pp. 177–223.

[22] G. Cannata and A. Sgorbissa, "A minimalist algorithm for multirobot continuous coverage," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 297–312, 2011.

[23] J. Bentley, "Programming pearls: algorithm design techniques," *Communications of the ACM*, vol. 27, no. 9, pp. 865–873, 1984.

[24] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Concorde TSP solver. [Online]. Available: http://www.tsp.gatech.edu/concorde

[25] G. Reinelt, "TSPLIB-A traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

[26] M. R. Garey and D. S. Johnson, *Computers and intractability*. San Francisco, CA: Freeman, 1979, vol. 174.

[27] S. Ntafos, "Watchman routes under limited visibility," *Computational Geometry*, vol. 1, no. 3, pp. 149–170, 1992.